CUTTING-EDGE TECHNOLOGIES

Computer Software: Strategic Industry

JOHN A. ALIC, JAMESON R. MILLER & JEFFREY A. HART

ABSTRACT *US-owned firms account for more than 60% of worldwide computer software sales, and US capability in software technology is widely viewed as well ahead of that in other countries. However, the US lead cannot be expected to last. Nor are policies yet in place intended to protect existing advantages. Japanese firms have demonstrated their capabilities in hardware. Japanese executives know they must develop better software in order (1) to use their own computer systems to best advantage, and (2) to sell more hardware in export markets. The next several decades will see a gradual slippage in the US position, particularly as foreign software suppliers move away from custom programming and related services, their present focus. A narrowing gap between US and foreign industries could prefigure a competitive challenge in software development not unlike earlier challenges in microelectronics. Better software, in addition, will have impacts elsewhere. In Japan's case, for example, improvements in software should lead to productivity enhancements throughout the economy, improving Japan's ability to compete internationally.*

Why Software is Critical

The cliché comes easily: information technology (IT) is to modern industrial economies what steel was to the industrial societies of the late 19th century, automobiles to the first half of the 20th. Computer hardware and software, telecommunications, embedded and invisible processors deep inside other equipment—these make it possible for banks to process enormous numbers of transactions quickly, to serve customers automatically, and to conduct business across international borders effortlessly and almost instantaneously. Computers help aeroplanes fly (the two dozen processors in a Boeing 767 implement half a million lines of code), and air traffic controllers to manage their movements. Computers and microprocessors run machine tools, control chemical processing operations, and schedule production operations, making factories more efficient and their products of higher quality.

The logic embodied in intricate and often lengthy programs—i.e. in software —tells the processors what to do. In a real sense, therefore, computing and

John A. Alic, Office of Technology Assessment, Washington, DC 20510, USA; Jameson R. Miller, Dean Witter Reynolds Inc., San Francisco, CA 94123, USA; Jeffrey A. Hart, Indiana University, Bloomington, IN 47405, USA. The views expressed are those of the authors, not those of the organizations with which they are associated. Parts of the paper draw on the Office of Technology Assessment report *International Competition in Services*, to which all three authors contributed; they thank Robert R. Miller and Kenan P. Jarboe for assistance.

information technology becomes a matter of software development more than hardware technology. Software embodies the logic of complex systems: because of this near-universal function, it is today's critical technology and critical industry *par excellence*—more so than chips, biotechnology or advanced materials. Software epitomizes high technology for the latter part of the 20th century, and increasingly will be the driving force in international competition among industrialized economies.

Both sales and R&D expenditures have been growing faster for software than for hardware. Software needs and availability shape the design of much computer hardware; indeed software is often integrated into hardware itself, as with functions embedded in semiconductor chips. With computer systems increasingly specialized and the industry increasingly fragmented (office automation, high-reliability systems, engineering workstations), functional capabilities created through software inevitably drive sales of broad ranges of equipment.

As a result, programming costs have become a larger proportion of total development costs for IT systems of all types. In fact, software-intensive computer-aided design methods are now necessary for developing new large-scale integrated circuits, while much programming itself depends on computer-aided software engineering techniques. In telecommunications, 80% or more of the multibillion dollar cost of a new family of programmable central-office (CO) switches goes for software. Companies producing CO switches can expect to spend several hundred million dollars annually simply to maintain and improve software, in particular for the software updates necessary for providing new services.

In a broad range of manufacturing industries, engineers have also come to rely on sophisticated computer-assisted methodologies for analysis and design of innovative products—e.g. finite element methods for predicting the behaviour of mechanical parts. When these products reach the factory floor, computer programs control the equipment used in processing, in shopfloor management, and in inspection and quality control.

Software structures the interactions between people and machines in organizations of all types, helping shape corporate routines, the contours of jobs, channels of power and influence. Computer-based systems help cut costs and create new industrial strategies throughout all advanced economies. Certainly for the USA, software tools will be vital in improving rates of productivity growth in both manufacturing and the services.

National security provides a final set of reasons for governments to focus on software development. During the 1950s and 1960s, the US computer industry gained its position of world leadership in large part because of spending by the Department of Defense (DoD). In those years, early warning systems, intended to detect possible attacks by aircraft or missiles, placed heavy demands on computer technology. Today, military equipment depends on advanced digital systems for missions ranging from fire-and-forget missiles to aircraft flight controls and ballistic missile defence.[1] In the future, defence systems will depend even more heavily on software. Military aircraft will incorporate pilots' aids that extend the capabilities of both man and machine by helping cope with information overload and flight manoeuvres beyond human skill levels. Huge data-intensive information systems such as those of the US National Security Agency create quite different but equally demanding requirements. Today, military computing and telecommunications rely to an increasing extent on technologies originating in

commercial industry, a trend expected to continue. In earlier years, technology frequently flowed in the other direction.[2] Maintaining competitive strength in civilian applications of computers and software will be vital for preserving national security in the future.

At the same time, while software has been growing more important for productivity, competitiveness, and national defence, efficiency in programming itself has been nearly stagnant. Software generation remains a highly labour-intensive activity, relatively more expensive as productivity improves elsewhere in the economy.[3] It follows that raising productivity in software development, by making better programs available more quickly and cheaply, holds enormous promise for improving efficiency throughout any industrial economy.

Today, in part because of the early impetus of US defence spending, US-based software companies can claim undisputed leadership in world markets, leadership that has given competitive strength to other US firms. Also, because of this derivative impact, other governments have viewed the US lead in software with a good deal of concern. They worry that lagging software capabilities could leave them dependent. As a result, foreign governments have been steering more funding to software R&D. Germany, for example, has begun to shift government support from hardware (e.g. microelectronics) to software. Even such newly industrializing countries as Singapore and Taiwan emphasize software in their government-sponsored programmes for catching up technologically. Also, of course, there are such well publicized efforts of the Japanese Government as its fifth-generation computer project, which sought, among other things, software that would help Japanese computer manufacturers penetrate world markets more effectively.[4]

The Software Bottleneck

Although price/performance ratios for hardware have been declining steeply for years, costs per line of software code remain about the same today as two decades ago.[5] This would imply that efficiency in programming has been improving at no more than national rates of productivity increase—in striking contrast to the extraordinary rates of improvement in hardware. The primary reason for the software bottleneck is a lack of unifying technical concepts and proven software engineering tools and methods. Highly skilled (and highly paid) employees must still write and debug software on a line-by-line basis; programming remains an extraordinarily labour-intensive activity. Although enhancements of computer languages and programming aids—including computer-assisted software engineering (CASE) tools—have helped, larger and more complicated programs continue to stretch the capabilities of the best people and the best tools.

In addition, maintenance—upgrades as well as debugging—typically accounts for well over half of life-cycle software costs.[6] Documentation is also expensive, with the spread of computing power to new and non-expert users making good documentation ever more important for success in the marketplace. The result? A productivity bottleneck in programming, with software now accounting for a far greater proportion of total system costs than in earlier years.

Given the dimensions of the bottleneck, it should be no surprise that a great deal of software R&D has been aimed towards tools for cutting costs and speeding common programming tasks. Much effort has gone into fourth-generation

languages intended to help users tailor software to their own needs, and artificial intelligence (AI) techniques for training programmers, as well as expert systems to help in their work.[7]

The US Defense Department has been deeply concerned with slow productivity improvement in software generation. As a consumer of software, DoD far outstrips the rest of the US government, spending some $30 billion annually for development, procurement and maintenance of computer programs.[8] Defence systems, moreover, harbour large numbers of incompatible computer systems, many of them highly specialized and running on the software equivalent of ancient languages. Avionics can account for half the cost of a modern military aeroplane, with 80% of this for the software (much of it embedded in, for instance, electronic warfare systems). Given the size of DoD's software expenditures, the inherent costs of incompatible systems, and of maintaining programs written in several hundred languages, most of them obscure, DoD has been trying to compel standardization on a single programming language, Ada.[9]

Beyond its efforts to implement Ada, DoD has sought software engineering advances aimed at increasing productivity and controlling costs, spending nearly $200 million annually on software research. For instance, the Department has instituted a programme called STARS (Software Technology for Adaptable, Reliable Systems) intended to push forward software engineering techniques, while also supporting software and systems engineering centres at universities. None of these efforts, however—including the push for Ada—seem likely to be more than modestly successful, for reasons that range from controversial choices of technology (Ada) to burdensome contracting provisions and lack of managerial and administrative focus in the huge Pentagon bureaucracy.

The US Software Industry

Although a rapidly growing independent software industry had emerged in the USA by the 1970s, computer hardware manufacturers—and users—continue to develop a great deal of software. Originally viewed as a marketing carrot for selling hardware, software has become a major source of revenue in its own right for computer manufacturers. Sales of software have been growing more rapidly than those of hardware (in part simply a consequence of slow productivity growth); profit margins have also tended to be greater. At the same time, independent firms develop and market programs for off-the-shelf sale or lease to customers—packaged software—along with customized programs tailored to users' specific requirements. Often, the suppliers provide training, documentation and at least some maintenance for their software products. Worldwide software revenues of US-based firms probably exceed $100 billion, accounting for more than 60% of the global market.[10]

In the early days of the computer industry, customers purchased hardware and software bundled as a package from one of the half-dozen or so companies that made computing equipment. Today, with literally thousands of firms, most very small, developing and distributing programs, software sales are split in roughly equal portions between hardware manufacturers and independent suppliers. However, sales by the independents have been growing faster—a trend that should continue.

Computer manufacturers mostly supply operating systems and applications

software, designed specifically for their machines, in addition to compilers, interpreters, utilities, and the like. This is a big market in terms of value: PCs vastly outnumber large systems, but programming the latter is many times more costly, and costs must be covered with far fewer unit sales. As a result, the software revenues of a number of the larger hardware manufacturers dwarf those of even the largest PC specialists. Microsoft, the biggest of the latter, billed about $975 million in 1989; IBM's software revenues totalled $8.4 billion.[11]

Many businesses continue to do some of their own programming—e.g. banks and accounting firms. However, for the lion's share of less specialized needs, the benefits of purchasing software from outside have become steadily more compelling. These advantages include the following:[12]

- *Cost.* Packaged, off-the-shelf software may cost 10 to 100 times less.
- *Availability.* Software already on the market can be quickly evaluated, purchased, and put to work.
- *Lower risk.* A firm choosing to develop its own programs may not achieve its functional goals; even if it does, experience shows that development will probably cost more and take longer than expected.
- *Flexibility and manpower savings.* Purchasing software minimizes specialized internal staffing requirements.
- *Better documentation.* Packaged software includes documentation, which can be evaluated before purchase. Few companies seem able to enforce high priorities for good documentation on in-house software projects.

Against these benefits, managers must weight the possibility of arriving at better functional solutions to their firm's particular problems—solutions that might lead to proprietary packages, sometimes a potent source of competitive advantage. Continued development of fourth-generation languages opens new opportunities for companies to get the advantages of proprietary software without the costs of full-custom programming.

The environment for software today is both highly competitive and technologically volatile. Some successful companies—Microsoft or Lotus Development—are still only a few years old, while older firms like Cullinet and a number of specialists in computer-aided design have encountered difficulties. High-end software specialists have sought to expand into other segments of the market, while also moving to exploit such technologies as fourth-generation languages. Although some kinds of routine software development can be handled by programmers with modest skills, market success often depends on the insights of a few unusually creative people—those who can understand what users of spreadsheets, database management systems, or computer graphics really want and need, develop expert systems (a form of AI), or make progress in automating the generation of software itself.[13] Mergers, acquisitions, and diversification have been commonplace, as firms try to expand their product lines, programming staffs, and customer bases. The larger user base resulting from falling hardware prices has led to software price declines, because suppliers can amortize development costs over larger unit sales.

Software Industries in Other Countries

With few exceptions, foreign firms lag well behind their US competitors in software technology and in sales. One reason is simply the large US market for IT

of all kinds. US companies have a better chance of covering their up-front design and development costs at home, giving them wider latitude in setting prices overseas. The large domestic market also lowers risks, compared with most other countries, since even a modestly successful product may sell enough copies to cover fixed costs. Thus, it should not be surprising to see foreign software firms investing heavily in the USA.

Japan's Software Suppliers

Although European software firms have been more visible internationally, over the long run Japan should emerge as the primary US rival in software. Japanese firms now manufacture computers that compare well technically with those from the USA, but international sales have been hampered by poor applications packages and limited sales and service networks.[14] In contrast, Japanese systems software, based on technology originating in the USA, is usually considered quite good.[15]

The Japanese recognize their deficiencies, and have embarked on a massive effort to catch up—indeed, to surpass the USA. A few years ago, for instance, Hitachi spent only 10% of its R&D money on software; now it allocates more than 30%. Toshiba has pioneered 'software factories' housing thousands of programmers to work on products for business and industry. Aside from such efforts, however, the Japanese software industry today resembles that in the USA perhaps two decades ago—small and not very visible. Independent software firms remain weak. Skilled programmers have been in short supply. At a time when customized programs have fallen below 35% of the US market, perhaps 90% of Japanese applications software continues to be undertaken on a custom basis, often internally.

This heavy reliance on custom programming is inefficient, and will not persist indefinitely; the burgeoning software needs of the Japanese economy can only be accommodated through greater adoption of standardized applications packages. Japan's rapidly growing hardware base, now second only to that of the USA, will force change, and, as they respond to these pressures, Japanese software suppliers will become more aggressive internationally, following the paths marked out in so many other industries.

In manufactured goods, Japanese firms have elevated process engineering to a high art. They are trying to do the same with programming. Software factories like Toshiba's reportedly produce high volumes of code with levels of quality (freedom from errors) and productivity (lines per man-year) substantially higher than in the USA. By specializing in particular applications—e.g. process-control packages for nuclear power plants or steel mills, aircraft flight controls—they can more easily re-use blocks of code, and train programmers narrowly but deeply.[16] Thirty per cent or more of a given package may be recycled from past programs, yielding improvements in both quality and productivity; Toshiba's Software WorkBench claims an error rate of 0.3 bugs per thousand lines of code, a factor of 10 below typical US error rates.

When the Japanese software industry moves, as it must, towards prepackaged applications programs, the software factory experience should be of considerable value. Indeed, this is part of the strategy: a Toshiba executive has said. 'To

overcome Japan's language problem and compete with the United States, we have to have productivity double that of the US.'[17] The Japanese expect to build strength in software just as they have in other industries—incrementally, by steady improvement of existing products and processes.

Of course, releasing a bug-free program means little if it fails to meet user needs. For general-purpose applications packages with vague and ill-defined requirements, market success depends first of all on conceptual design. Here, Japanese software firms generally lack experience (despite the publicity given 'fuzzy logic' in Japan). However, if the ideas become available—perhaps from US companies or US software designers hired by Japanese firms—Japan's experience base could provide a foundation for future productivity and cost advantages. To surmount their handicaps in conceptual design, Japan's software suppliers will not hesitate to follow semiconductor and automobile firms in establishing design centres in the USA, or in licensing US inventions (as in Toshiba's recent move into workstations using Sun Microsystems' SPARC technology).

Japan's difficulties, many of them based on language differences, mask some real strengths. Efforts to develop Japanese language input-output terminals, and, more recently, word processing software, provide a useful foundation for some kinds of applications packages. Programmers in Japan, not surprisingly, prefer to work in their own language. To export software, they must translate not only code (commands, prompts and comments) into other languages, but also the accompanying documentation and training materials—an area of particular weakness. Japanese success in developing kanji-based word processing programs, however, implies significant advances. These systems interpret keystrokes representing phonetic combinations, 'guessing' the operator's meaning based on context and expressing that meaning in kanji. Programs that accomplish this become, in effect, applications of AI, with potential for transfer to other types of programs and obvious competitive implications.[18] In manufacturing, finally, Japanese companies have implemented simple but sophisticated factory automation systems, with software already well proven in practical applications: Japanese software for numerically controlled machine tools, for automated inspection, and for statistical quality control may be less than innovative—perhaps even derivative of US technology—but it works, and works well.

Thus far, however, it is the fifth-generation project that has attracted most of the attention in the West. In progress since 1982 under the auspices of the Institute for New Generation Computer Technology (ICOT), the original goal of the fifth-generation project was to extend applications of massive computing power to ordinary users by harnessing AI, natural language input capability, and very large databases. The Japanese hoped to leapfrog existing (i.e. US) computer technologies by turning joint government-industry R&D—a process refined in Japan over several decades—to the 'software gap'.[19] A good deal of scepticism has been heard in the West (and in Japan) concerning the technical directions chosen by ICOT. However, as many other joint R&D efforts in Japan have demonstrated, focusing exclusively on technology misses the point. These projects serve many other functions in Japan's industrial policy system, ranging from consciousness raising and consensus building to training technicians and engineers. In Japan, concerted effort to build an 'information economy' goes back to the 1960s; the government looks to computers and communications as the centrepiece of the nation's future economic structure, one emphasizing knowledge-intensive, hence software-intensive, goods and services. Individual projects

like the fifth-generation are only one part of this much larger push, and the Japanese have already begun to discuss a sixth-generation successor project.

European Industry

Although custom software does not take as large a fraction of sales in Europe as in Japan, it remains more common than in the USA. Within Europe, Germany is the largest market, followed by the UK and France. If third as a market, France nonetheless has the strongest software industry in Europe. Cap Gemini Sogeti, for instance, the largest European software firm, has been very aggressive in seeking growth through acquisitions; the company does more than half its business outside France. French (and German) computer hardware and telecommunications equipment manufacturers have also been major players in software markets, as in the USA and Japan.

At the same time, such US-based computer firms as IBM, Digital Equipment and Hewlett-Packard have substantial presences in Europe, both in hardware and software. IBM operates half a dozen European R&D centres, each undertaking some work related to software. The firm has held about a quarter of the Western European market for off-the-shelf systems software in recent years; indeed US-owned firms account for most of the packaged systems programs sold in Europe, although European suppliers do better when it comes to custom software. Some of the independent US firms have invested in European subsidiaries, on occasion conducting R&D abroad. ADP's Dutch subsidiary, for example, developed software for auto parts wholesalers and retailers that is now marketed through ADP offices elsewhere in Europe.

As in the USA and Japan, the fastest-growing portion of Europe's software market consists of PC applications packages. European businesses lag significantly behind the USA in PC purchases, even though about as many people work in offices there. Thus it is no surprise that, from their beginnings, US-based specialists in PC software have sought and found markets in Europe. Today, firms like Microsoft and Lotus supply software packages in all the major European languages.

European software suppliers, including the computer manufacturers, remain minor players outside the continent. As in Japan, however, governments have been funding R&D intended to strengthen capabilities in software—through pan-European programmes such as ESPRIT (European Strategic Programme for Research in Information Technology), funded by the European Community since 1984, as well as national efforts.[20]

What Should the USA Do?

Leadership can easily breed complacency. Many Americans seem to think that software is somehow special—that countries like Japan will be unable to succeed as they have in other technologies and other industries. This view is wrong. The software industry is headed for the same kind of trouble as the semiconductor industry. US firms may be the innovators, but software development will continue to demand painstaking attention to detail. Japanese firms have amply demonstrated their skills in high-quality, error-free products in other industries. Already, they have begun applying these skills to software. And even without new

competitive threats, the USA will have to take steps to deal with its domestic software bottleneck, the consequence of low productivity growth.

The overall problem is much the same as in earlier competitive challenges. US industry is not slipping. Rather, foreign firms are improving. As the software industry comes under pressure from abroad, US managers will eventually troop to Washington and political figures will take up the cause. If past experience is any guide, there will be little consensus on the nature of the problem, or on remedies. More than likely, several years will pass in debate with little resulting action. This has all happened before, notably in semiconductors, where a decade of trade friction with Japan preceded the effort, through the R&D consortium Sematech, to address the technological dimensions of the problem.

For the USA, the only way to avoid repeating this story is to begin investing in stronger technological foundations in software, so that US firms can continue to move down the learning curve ahead of their rivals overseas. If there is one lesson that the history of the past two decades should teach, it is the folly of waiting until an industry is in competitive difficulty before beginning the policy debate. So far, US software suppliers have been content to seek stronger protection for intellectual property to help safeguard their edge. This is a slender reed. It cannot hurt, but will do little to protect the existing lead.

Intellectual Property Protection

Computer programs are expensive to develop and cheap to produce, inviting illegal copying. Technical solutions intended to make copying impractical or impossible have not worked; software pirates quickly find ways around new protective schemes, no matter how ingenious, just as thieves manage to keep stealing cars. However, should not stronger protection for intellectual property, particularly in foreign markets, encourage innovation and safeguard the US lead? In fact, matters are not so simple.

First of all, software, in effect, falls between the stools of copyright and patent law. Copyrights protect code but not the function or the logical structure of the program. Because the same functions can be coded in many different ways, protecting only the code accomplishes little. Software patents protect the functional aspects of a progam—what it does. Originally, software patents in the USA were limited by court decision to programs that implemented physical processes —e.g. controlling a typesetting machine. More recently, the trend has been to grant patents even if the link to processes is tenuous.[21]

Although patent protection has become a more effective means of protecting software in the USA, intellectual property law in much of the rest of the world has hardly begun to confront such questions. The laws in many countries are weak, and enforcement lax. Some governments ignore copying and counterfeiting (of many products, not just software). Given the slow pace in multilateral forums like the World Intellectual Property Organization, the USA has negotiated bilaterally with countries including South Korea, Thailand, and Japan.

These negotiations have made a difference, but the real question is this: What would more effective intellectual property protection, in the USA and abroad, accomplish for the US software industry? Traditionally, the intent has been viewed as encouraging innovation through a limited monopoly that rewards those who pioneer new technology. Indeed early in this century, patents helped companies like General Electric and AT&T dominate their industries. However,

those days are gone. Big business in the USA may still reflect patterns that emerged in the era of trusts and trust-busting, but in recent years the competitive significance of patent protection has been restricted mostly to the chemical and pharmaceutical industries.[22] Intellectual property is not irrelevant to the evolution of computer technology, but it has been a sideshow. Should the US government, which has had no strategy for software other than to seek stronger protections for intellectual property, decide to develop one, that strategy should aim at breaking the productivity bottleneck and creating conditions under which US companies can improve their innovative capabilities more rapidly than foreign firms.

Technology Policies

The fundamental problems in software are technological, stemming from slow growth in productivity in programming itself. The Japanese strategy begins with recognition of the bottleneck, seeking to break it through industrialization of programming and improved quality. From the US perspective, a strategy for supporting software technology and the software industry might include the following elements.

(1) *Substantial expansion of R&D funding* aimed at building the research and technology base for software. Program generation will remain an art, but better tools could improve efficiency and help US firms remain ahead. One route to greater productivity in programming is to put software on a sounder theoretical footing. R&D support should come both from DoD, which has been putting much emphasis on CASE tools, and the National Science Foundation (NSF).

Many US software firms have been caught in the same trap as US semiconductor firms: putting all their resources into designing next year's products, to the neglect of the technology base. Smaller firms, in particular, may have no choice; this is their only hope for keeping up in a rapidly expanding market. However, lacking a broad and deep technology base, US firms will find, as the industry matures, that they do not have the tools to compete with far-sighted foreign firms that have built carefully for the future.

(2) *A government-wide mandate for purchase of standardized, off-the-shelf software*, instead of the expensive and often ineffective custom programming that agencies frequently want. Supporting the US industry through procurement would reinforce the market forces that already exist.

Private companies have learned that standardized, off-the-shelf programs can meet many of their needs. They can do so for government agencies too, but the agencies, lacking the discipline of economic competition, do not always realize it. Packaged software will be the competitive battleground of the future; simply by buying commercially available programs, the US government could support those portions of the domestic industry that will face the most intense competition in the years ahead. At the same time, the government could save money by buying off-the-shelf. Finally, defence procurements should be simplified to attract smaller software firms lacking experience in negotiating the bureaucratic maze of DoD contracting practices.

(3) *Aggressive support for training at all levels, from technicians to graduate-level software engineers.* The software industry grew by attracting people from other sectors of the economy. As yet, there is little of the sense of professional identity

that gives focus to older engineering disciplines, and helps create a thriving professional community.

Well trained people will become an even more vital resource for US software firms in the future, as the field continues to mature. The problems begin in the primary and secondary schools, where the foundations for later study must be laid.[23] At the university level, the need is for courses and curricula that focus on the design of complex software packages that satisfy user needs. Much of this work is closer to engineering design and to the behavioural sciences than it is to the applied mathematics emphasis of current academic computer science programs. More of a focus on software as a professional discipline will be needed to attract students and provide a large enough pool of well-qualified people for US firms to draw from in the years ahead. Both NSF and DoD might seek to rapidly increase their support for graduate students in software engineering, for state-of-the-art hardware and software for teaching and research purposes, and for curriculum development in order to provide the legitimacy needed for a secure academic future. Government agencies should also provide expanded support for intermediate-level software training in community colleges.

The steps outlined above will need to be complemented by continuing efforts to secure better intellectual property protection in overseas markets, and to keep those markets open for US firms. However, the imperative in software is to break the productivity bottleneck—a technological problem, not a trade problem.

Conclusion

Software is critical for future economic growth and competitiveness. Productive uses of digital systems depend on software, giving it significance for the creation of wealth going far beyond its direct impacts. Taking advantage of a substantial lead time and large domestic markets, US-based companies have long been dominant in international competition. In the future, however, this dominance seems bound to wane, perhaps slowly but none the less inexorably.

Japanese and European industries, where custom programming is still the rule, will be forced, sooner or later, to move aggressively into the design and production of more standardized software—the major US strength. Specially tailored software is expensive, and no longer a good solution to many customer needs. Cost pressures will drive suppliers in other countries—especially Japan—towards the standardized applications packages pioneered by US firms. As foreign software companies begin producing these standardized programs, they will more effectively confront long-dominant US suppliers; the Japanese, in particular, will become more competitive simply because of the imperatives created by their rapid progress in computer hardware and their competitive strategies for exploiting this hardware. As foreign firms negotiate this transition, some will emerge better able to compete with US suppliers.

With software increasingly influencing the design of hardware, the Japanese, in particular, realize they will need major advances in programming capabilities as they move towards an information-centred economy. Both government and industry in Japan have made impressive commitments to improving productivity in programming and to new generations of software technology. The implied objective is to leapfrog the USA in this seminal technology. Japanese gains will stem in part simply from a comparatively weak position at the start of real

competition, in part from the strength of Japan's electronics industry as a whole. Fifteen years ago, Japanese companies were behind in computers and semiconductors; concerted efforts, not unlike those now under way in software, led to steady growth and greater competitiveness.

In the face of these ongoing efforts to catch up, in Europe as well as Japan, US software firms cannot expect to maintain their current share of the global market. Still, with the interdependences between hardware and software design increasing, the enormous domestic market base will be a continuing source of strength for the US industry. So also with other traditional sources of US advantage, including a talented workforce, strong in the conceptualization of complex programs and in grasping new user needs, and substantial R&D spending from both private and public sources, including defence programmes.

When it comes to government policies, however, the USA seems in danger of repeating past mistakes. Few in the USA seem as yet to realize the true significance of software. Trade friction with Japan in microelectronics, the heavily publicized Sematech venture, and the subsequent proposal for a joint venture to manufacture memory chips, have kept the spotlight on hardware. With policy makers complacent when it comes to the software lead, the USA risks losing advantages in a technology even more vital than microelectronics.

Notes and References

1. *SDI: Technology, Survivability, and Software* (Washington, DC, Office of Technology Assessment, May 1988), especially Chapter 9.
2. For example, both local and wide area computer networks, now rapidly diffusing through the civilian economy, utilize technologies stemming from the US Defense Department's ARPANET system, developed in the 1960s. On the military roots of the US computer industry, see *International Competitiveness in Electronics* (Washington, DC, Office of Technology Assessment, November 1983), pp. 81–92 and 145–148; and K. Flamm, *Targeting the Computer: Government Support and International Competition* (Washington, DC, Brookings Institution, 1987).
3. W. J. Baumol, S. A. Batey Blackman & E. N. Wolff, 'Unbalanced growth revisited: asymptotic stagnancy and new evidence', *American Economic Review*, 75, 1985, p. 806.
4. J. A. Alic & R. R. Miller, 'Export strategies in the computer industry: Japan and the USA', *Technology Analysis and Strategic Management*, 1, 1989, p. 11.
5. These costs range from about $10 per line for routine programs, to as much as $1000 per line for applications extraordinarily demanding in terms of freedom from errors ('bugs'), reliability, and fault-tolerance—e.g. for the space shuttle. E. J. Joyce, 'Is error-free software achievable?', *Datamation*, 15 February 1989, p. 53. Average programmer productivity—which varies by 10–20 times or more among individuals—has been increasing at no more than 5–10% per year.
6. Some estimates indicate that as much as 80% of maintenance costs go towards adapting software to customer needs that were not fully understood when the development process began, or that changed later. J. Martin and C. McClure, *Software Maintenance: The Problem and Its Solution* (Englewood Cliffs, NJ, Prentice-Hall, 1983). For an idea of the scope of maintenance requirements, note that the worldwide inventory of programs written in COBOL, still popular for business applications reaches 75 *billion* or more lines of code. On the general problem, see F. Brooks, 'No silver bullet: essence and accidents of software engineering', *IEEE Computer*, April 1987, p. 10.
7. With a third-generation language like BASIC or COBOL the programmer writes instructions that tell the computer in step-by-step fashion how to proceed. With a fourth-generation language—more properly termed an applications package—the

programmer tells the system what the output should be, but not how to achieve that output. End users may be able to create their own applications programs, combining the advantages of off-the-shelf purchases with those of a custom package. See Martin & McClure, *op. cit.*, Ref 6, chapter 11; also J. Martin, *Fourth Generation Languages* (Englewood Cliffs, NJ, Prentice-Hall, 1985).

8. D. Hughes, 'Computer experts discuss merits of defense dept. software plan', *Aviation Week & Space Technology*, 16 April, 1990, p. 65.
9. J. Voelcker, 'Ada: from promise to practice?' *IEEE Spectrum*, April 1987, p. 44; R. L. Hudson, 'Inventor's hopes still high for his computer language', *Wall Street Journal*, 24 May 1989, p. B2.
10. M. V. Vasilik, D. R. Woodward & M. P. Galen, *Survey of the software industry*, M90–26 (Bedford, MA, The MITRE Corporation, 1990); *1989 US Industrial Outlook* (Washington, DC, Department of Commerce, January 1989), p. 26–3. About a third of the software sales of US firms come from overseas markets.

    The MITRE estimate, for 1988, breaks down as follows:

Packaged Software	
Independent software firms	$ 10 billion
IBM and other hardware suppliers	16
Custom Programming	
Commercial	18
Defence	8
Embedded Software	
Commercial	15
Defense	18
Other (including telecommunications, education, and entertainment)	15
Total	$100 billion

11. 'Software's big 50', *Datamation*, 1 December 1990, p. 67; 'The Datamation 100', *Datamation*, 15 June 1990.
12. W. L. Frank, *Critical Issues in Software* (New York, Wiley-Interscience, 1983), p. 166.
13. Little has been written on the division of labour in software generation, but see Philip Kraft, *Programmers and Managers: The Routinization of Computer Programming in the United States* (New York, Spinger-Verlag, 1977).
14. Alic & Miller *op. cit.*, Ref. 4.
15. H. J. Welke, *Data Processing in Japan* (Amsterdam, North-Holland, 1982), Chapter 6; D. Brandin, *et al., JTECH Panel Report on Computer Science in Japan* (La Jolla, CA, Science Applications International Corporation, under contract no. TA–83–SAC–02254 from the US Department of Commerce, December 1984), p. 3–1. Also see M. V. Zelkowitz, *et al.*, 'Software engineering practices in the US and Japan', *IEEE Computer*, June 1984, p. 57. Both Fujitsu and Hitachi continue to make IBM-compatible computers, while NEC's operating systems trace their ancestry to Honeywell products. These operating systems may have origins in US technology, but today in at least some cases the Japanese versions are superior. One of the objectives of Japan's fifth-generation computer project was to help Japanese companies take the next step in breaking free of their remaining dependence on US software.
16. The specialities given are those of the Software WorkBench of Toshiba Fuchu—Brandin *et al., op. cit.*, Ref. 15, pp. 3–3 to 3–4.
17. 'Software: the new driving force', *Business Week*, 27 February 1985, p. 98.
18. Sales of Japanese-language word processors—mostly dedicated machines rather than PCs—took off in 1985; average prices dropped by a factor of five, and output rose from about 200,000 units to nearly 1 million. Production doubled again the next year, and by 1989 about 2.8 million word processors were sold in Japan, plus 1.6 million PCs. H. Morita, 'Character processing', *Business Tokyo*, July 1988, p. 27;

K. Mori & T. Kawada, 'From kana to kanji: word processing in Japan', *IEEE Spectrum*, August 1990, p. 46.

19. On Japan's approach to joint government-industry R&D, including the objectives of the fifth-generation project, see Alic & Miller, *op. cit.*, Ref. 4. One of the more important of the earlier efforts, the Pattern Information Processing System (PIPS) project, helped Japanese companies develop technologies for input devices that could accept kanji characters. H. Nishino, 'PIPS (Pattern Information Processing System) project—background and outline', *Proceedings of the 4th International Joint Conference on Pattern Recognition*, 7–10 November 1978, Kyoto, Japan (International Association for Pattern Recognition, 1978), pp. 1152–1161. The more recent SIGMA project (Software Industrialized Generator and Maintenance Aids), initiated in 1985 by the Information Technology Promotion Agency, has had more modest-seeming but just as noteworthy objectives: increased productivity in programming through software engineering techniques and automation. This effort, with a five-year budget totalling about $200 million and involving more than 175 companies, was originally scheduled to end in 1989, but has been extended. 'Sigma Project: its status and future prospect', Report Memorandum #148, Tokyo Office of the US National Science Foundation, 9 February 1988.
20. In mid-1988, despite many voices urging that the UK's Alvey Programme be continued, the government declined to extend it, stating that worthy projects should begin competing for R&D funds under ESPRIT. S. Watts, 'Alvey epilogue looks to the future', *New Scientist*, 14 July 1988, p. 4I; also Advisory Council for Applied Research and Development, *Software: A Vital Key to UK Competitiveness* (London, Her Majesty's Stationary Office, 1986).

    A number of developing nations are also seeking to build viable software industries, notably India, Singapore, Taiwan, and Brazil. India has perhaps been most aggressive, seeking to exploit its large pool of chronically underemployed university graduates. K. K. Sharma, 'India signs up the world for its software', *Financial Times*, 23 February 1989, p. 3. Several US companies, including Texas Instruments and Citicorp, have established software development facilities in India, while others have contracted out programming to local firms. Singapore and Taiwan have also sought to establish themselves as centres for software development, but have faced acute shortages of trained people. See, in general, Thierry Noyelle, 'Computer software and computer services in India, Singapore, the Phillippines, Hong Kong and the Republic of Korea', Report to UNCTAD, Conservation of Human Resources, Columbia University, July 1990.

    Brazil's approach has been different—an infant industry strategy relying on trade barriers to keep out foreign firms, thus reserving the market for local suppliers, including manufacturers of computer hardware and telecommunications equipment. For a broad review of Brazilian policies, see *Transborder Data Flows and Brazil* (New York, United Nations Centre on Transnational Corporations, 1983); also C. Frischtak, 'Brazil', in F. W. Rushing & C. G. Brown (eds), *National Policies for Developing High Technology Industries* (Boulder, CO, Westview. 1986), p. 31.
21. W. M. Bulkeley, 'Will software patents cramp creativity?', *Wall Street Journal*, 14 March 1989, p. B1.
22. R. C. Levin, A. K. Klevorick, R. R. Nelson & S. G. Winter, 'Appropriating the returns from industrial research and development', *Brookings Papers on Economic Activity, 3*, 1987, p. 783.
23. In tests of 13-year-olds, administered during 1988 by the Educational Testing Service to random samples of about 1000 students from five countries, American students ranked last in mathematics and fourth in science. A. E. Lapointe, N. A. Mead & G. W. Phillips, *A World of Differences: An International Assessment of Mathematics and Science* (Princeton, NJ, Educational Testing Service, 1989).